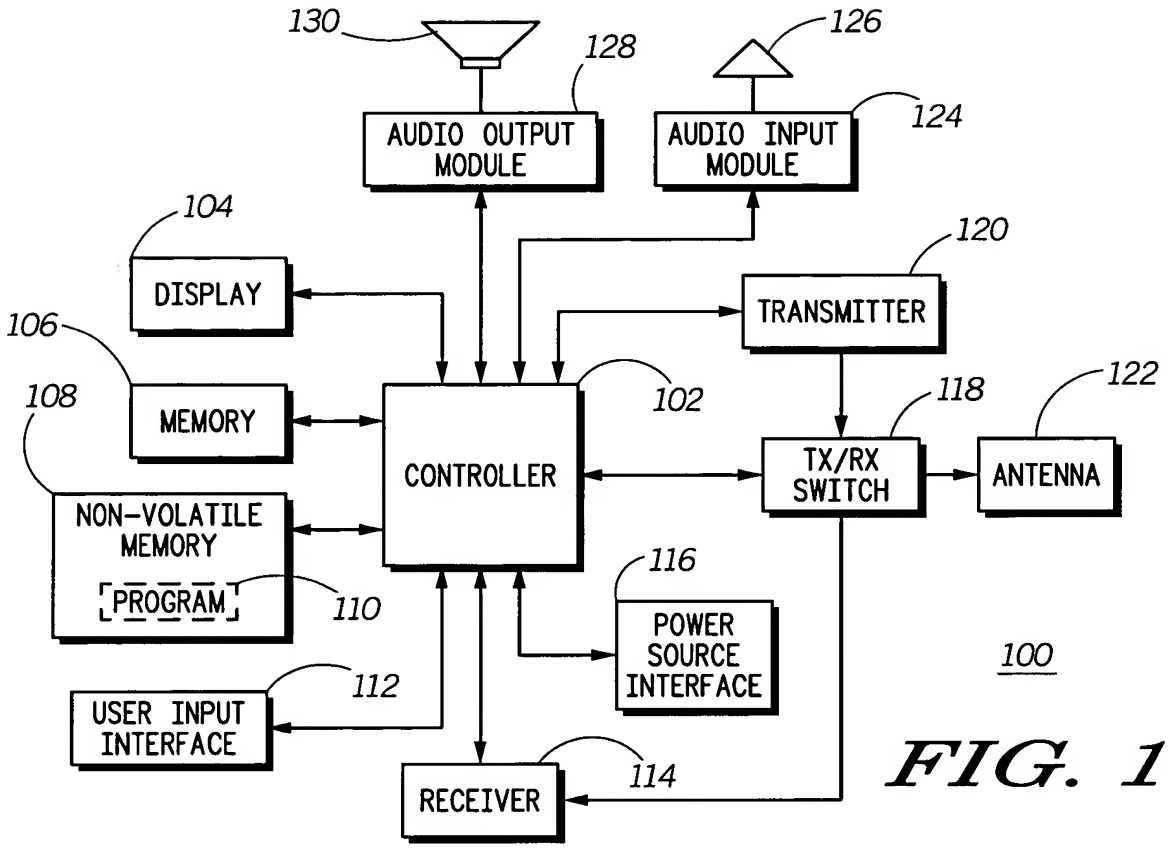

130
128
126
AUDIO OUTPUT MODULE
AUDIO INPUT MODULE
124
104
DISPLAY
120
TRANSMITTER
106
108
MEMORY
102
118
122
CONTROLLER
TX/RX SWITCH
ANTENNA
NON-VOLATILE MEMORY
PROGRAM
110
116
POWER SOURCE INTERFACE
100
112
USER INPUT INTERFACE
RECEIVER
114

FIG. 1

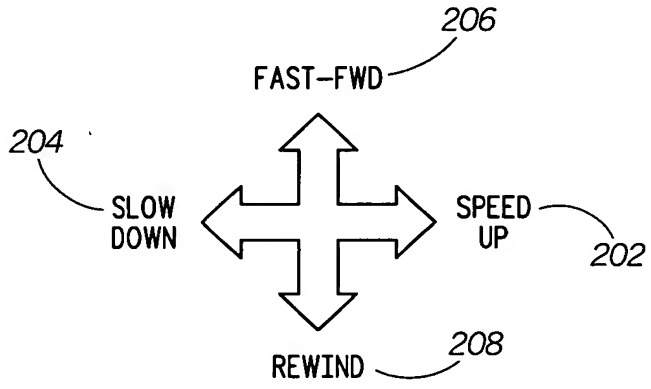

206
FAST-FWD
204
SLOW DOWN
SPEED UP
202
REWIND
208

FIG. 2

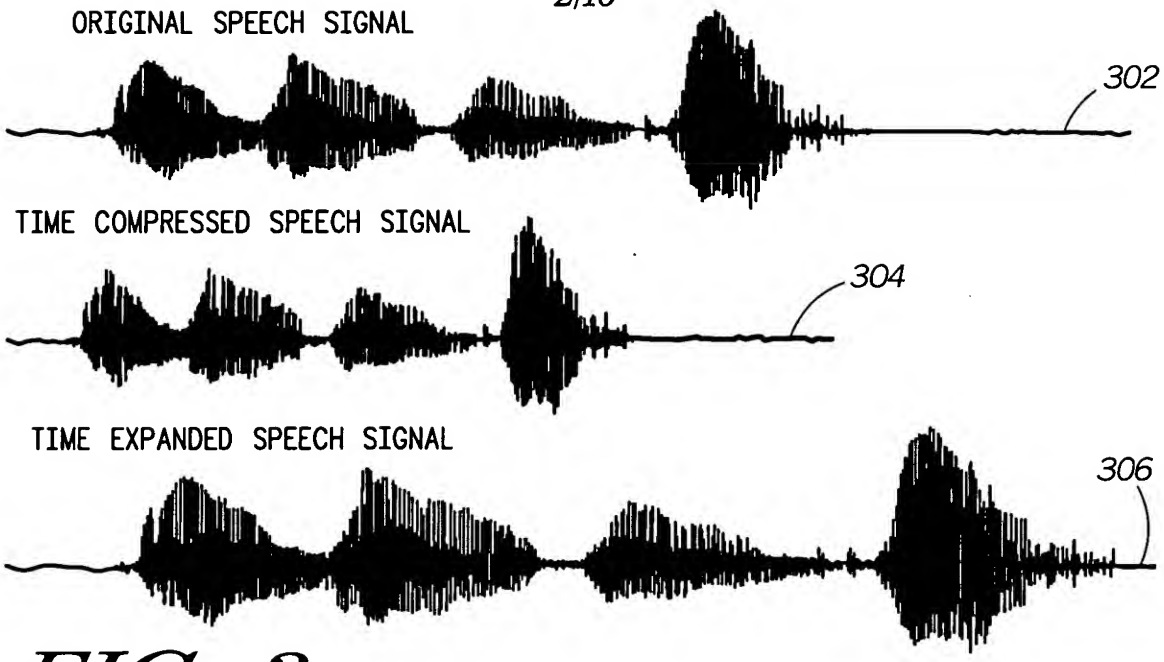

ORIGINAL SPEECH SIGNAL
302
TIME COMPRESSED SPEECH SIGNAL
304
TIME EXPANDED SPEECH SIGNAL
306

FIG. 3

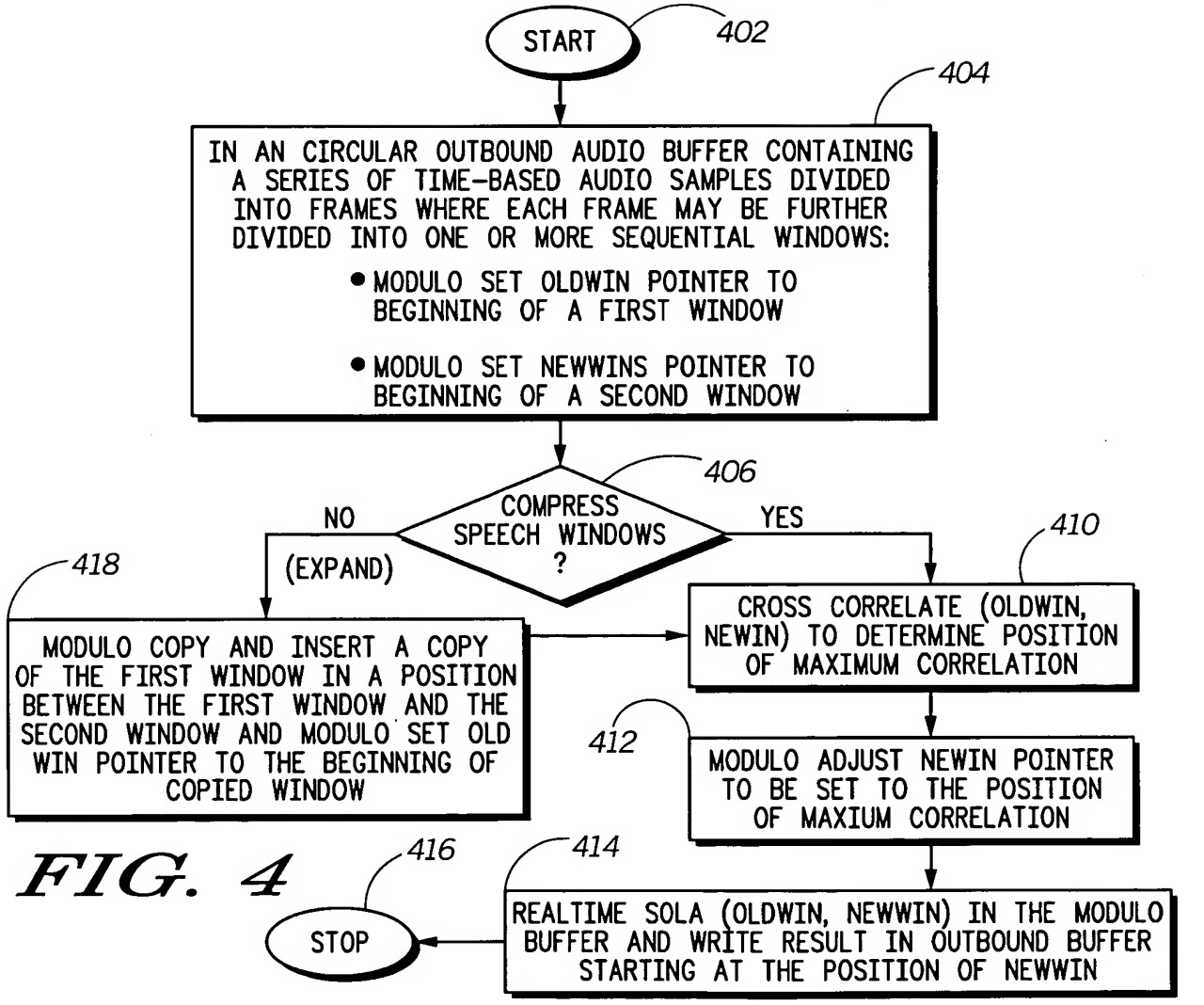

START
402
404
IN AN CIRCULAR OUTBOUND AUDIO BUFFER CONTAINING A SERIES OF TIME-BASED AUDIO SAMPLES DIVIDED INTO FRAMES WHERE EACH FRAME MAY BE FURTHER DIVIDED INTO ONE OR MORE SEQUENTIAL WINDOWS:
• MODULO SET OLDWIN POINTER TO BEGINNING OF A FIRST WINDOW
• MODULO SET NEWWINS POINTER TO BEGINNING OF A SECOND WINDOW
406
COMPRESS SPEECH WINDOWS ?
NO
(EXPAND)
YES
410
418
MODULO COPY AND INSERT A COPY OF THE FIRST WINDOW IN A POSITION BETWEEN THE FIRST WINDOW AND THE SECOND WINDOW AND MODULO SET OLD WIN POINTER TO THE BEGINNING OF COPIED WINDOW
CROSS CORRELATE (OLDWIN, NEWIN) TO DETERMINE POSITION OF MAXIMUM CORRELATION
412
MODULO ADJUST NEWIN POINTER TO BE SET TO THE POSITION OF MAXIUM CORRELATION
414
416
STOP
REALTIME SOLA (OLDWIN, NEWWIN) IN THE MODULO BUFFER AND WRITE RESULT IN OUTBOUND BUFFER STARTING AT THE POSITION OF NEWWIN

FIG. 4

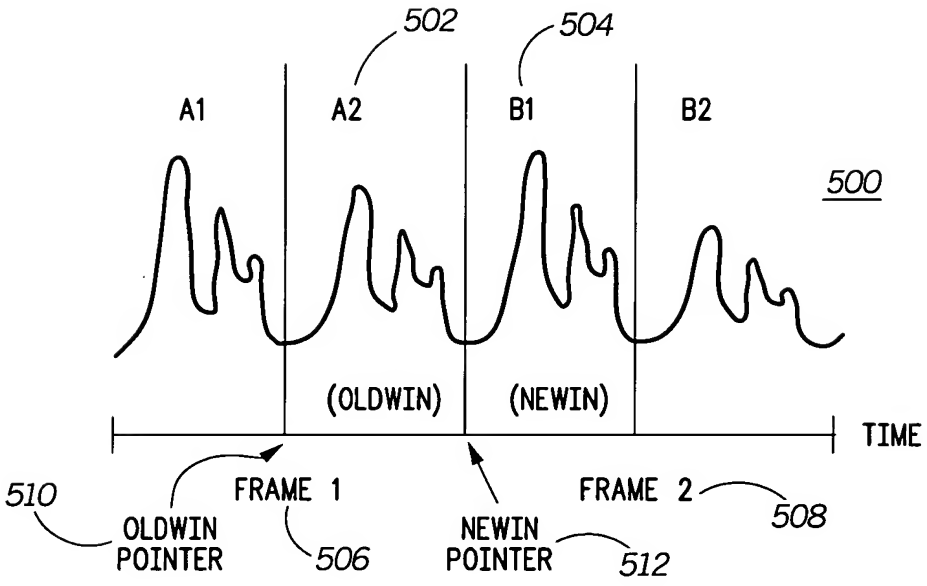

502
504
A1
A2
B1
B2
500
(OLDWIN)
(NEWIN)
TIME
510
FRAME 1
FRAME 2
508
OLDWIN
POINTER
506
NEWIN
POINTER
512

FIG. 5

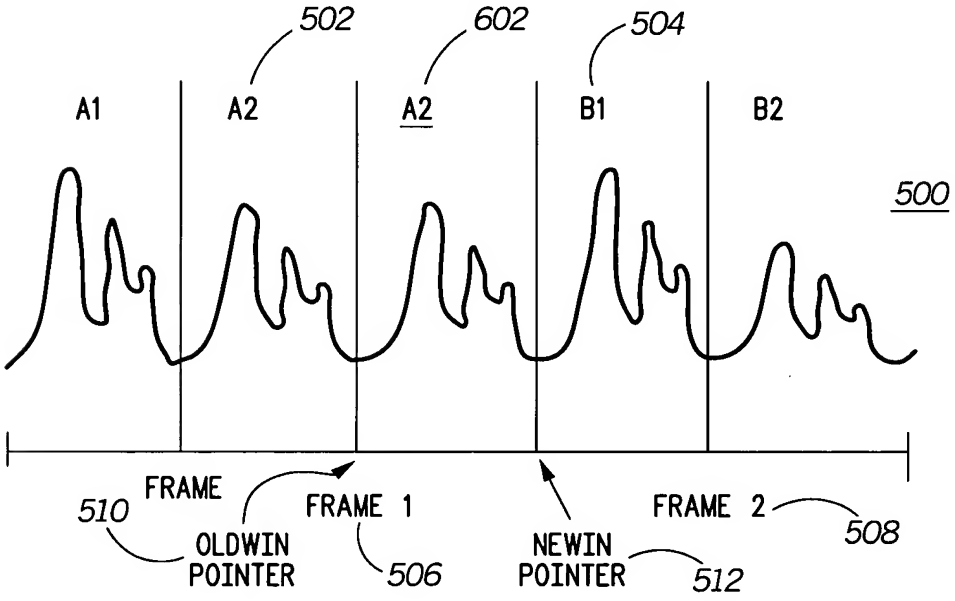

502
602
504
A1
A2
A2
B1
B2
500
FRAME
510
FRAME 1
FRAME 2
508
OLDWIN
POINTER
506
NEWIN
POINTER
512

FIG. 6

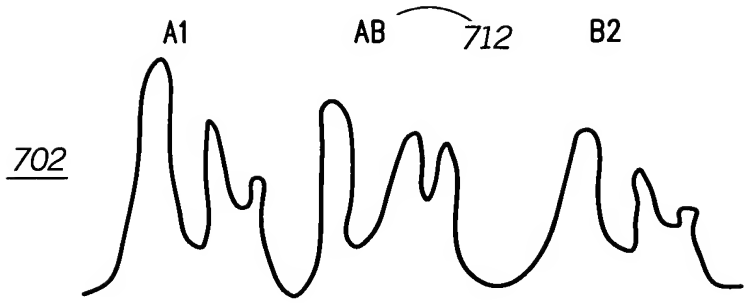
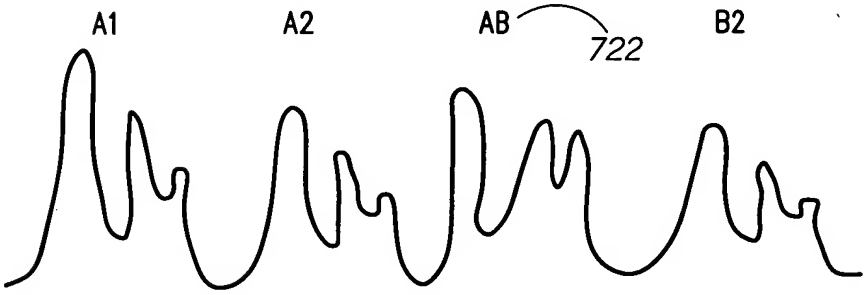

A1
AB
712
B2
702

FIG. 7


A1
A2
AB
722
B2
704

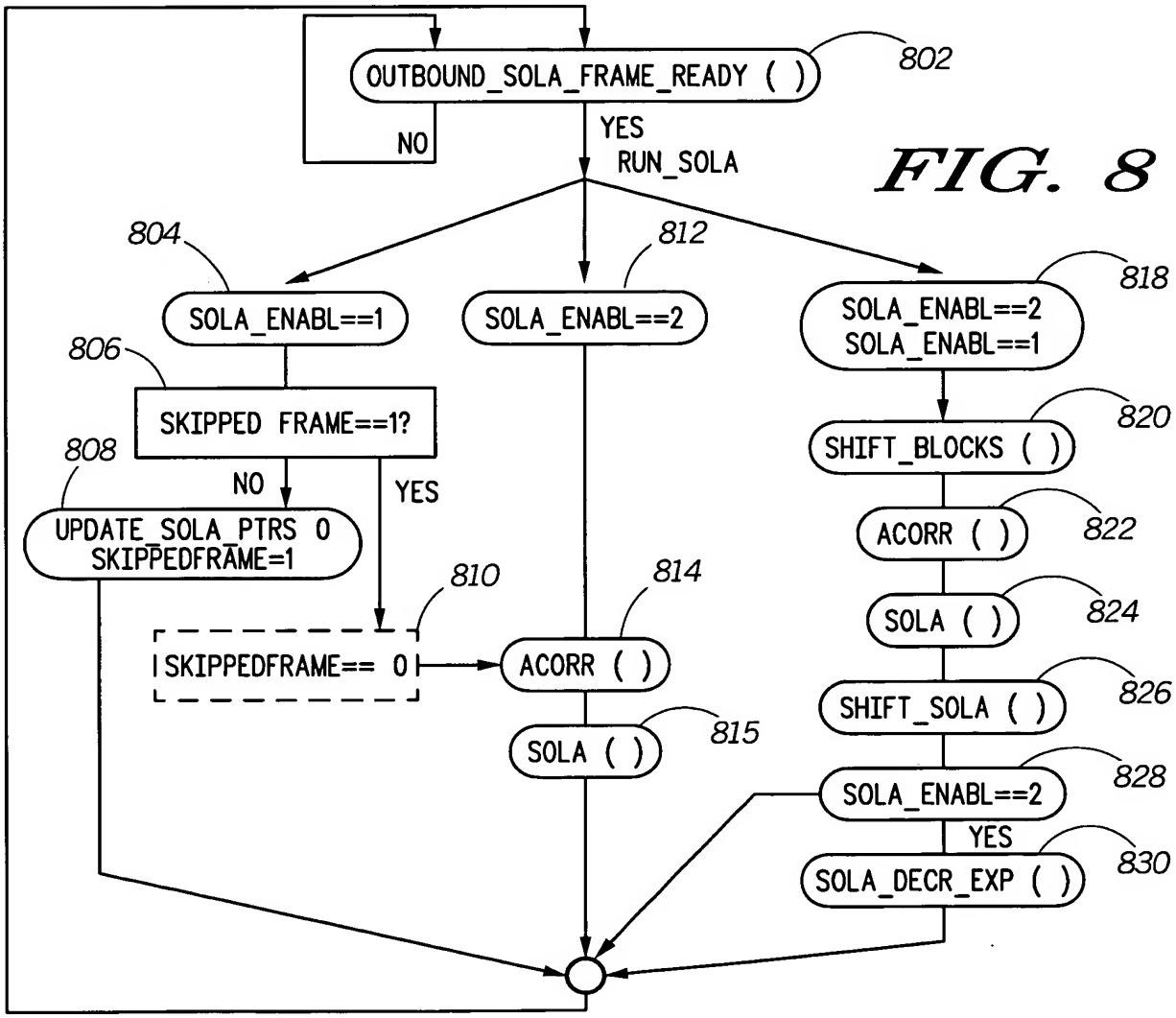

OUTBOUND_SOLA_FRAME_READY ( )
802
NO
YES
RUN_SOLA
804
SOLA_ENABL==1
812
SOLA_ENABL==2
818
SOLA_ENABL==2
SOLA_ENABL==1
806
SKIPPED FRAME==1?
820
SHIFT_BLOCKS ( )
808
NO
YES
UPDATE_SOLA_PTRS 0
SKIPPEDFRAME=1
ACORR ( )
822
SOLA ( )
824
810
814
SKIPPEDFRAME== 0
ACORR ( )
826
SHIFT_SOLA ( )
SOLA ( )
815
828
SOLA_ENABL==2
YES
830
SOLA_DECR_EXP ( )

FIG. 8

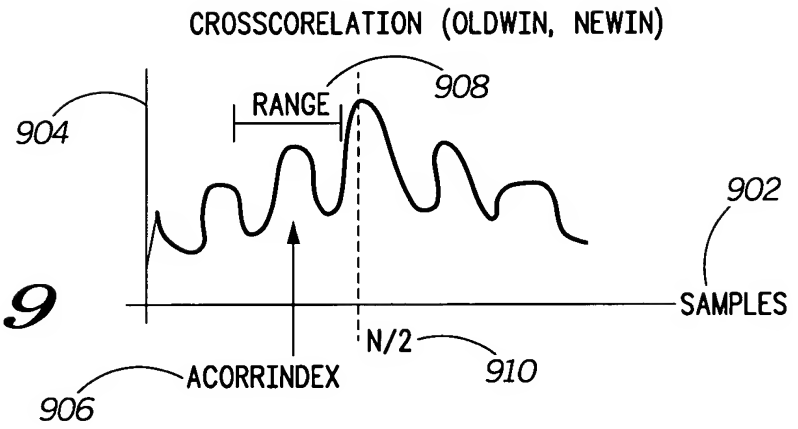


FIG. 9

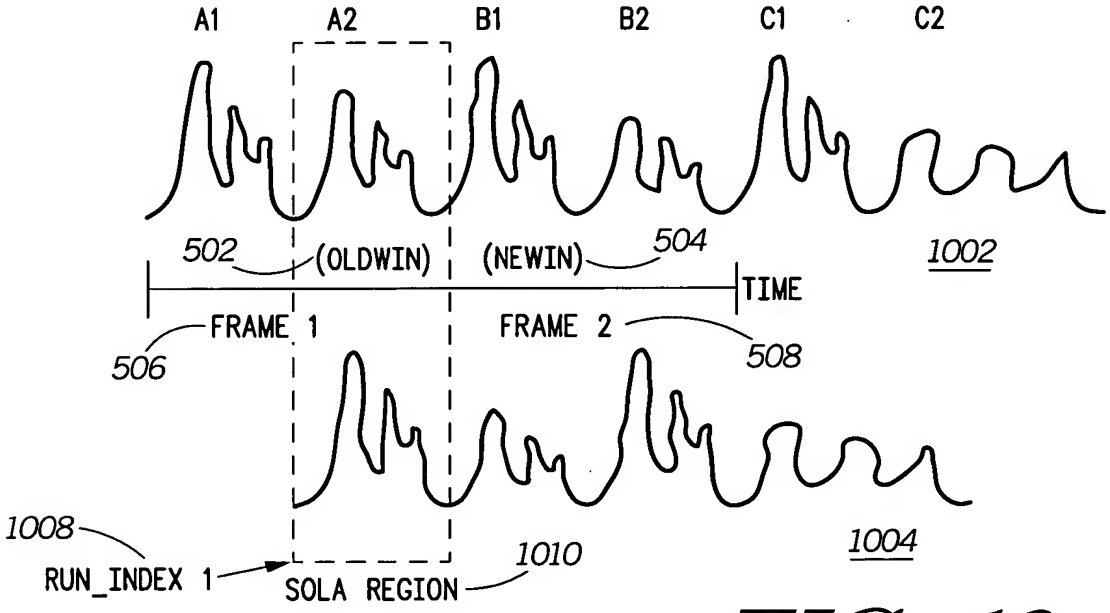


FIG. 10

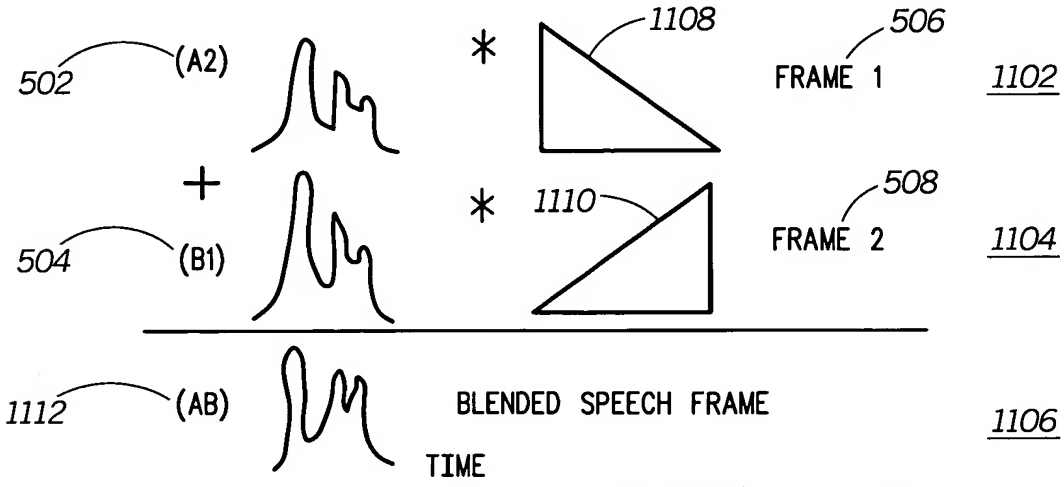


FIG. 11

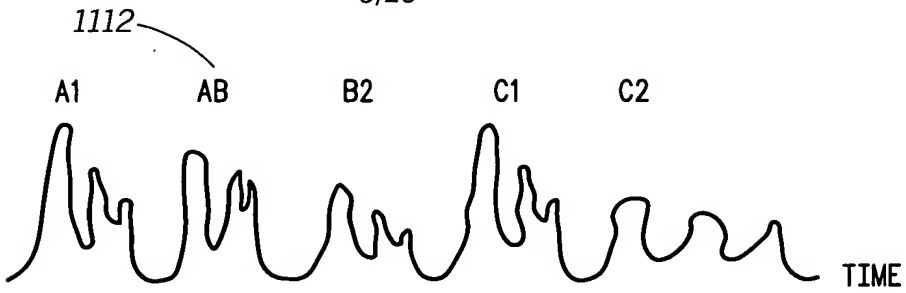


FIG. 12

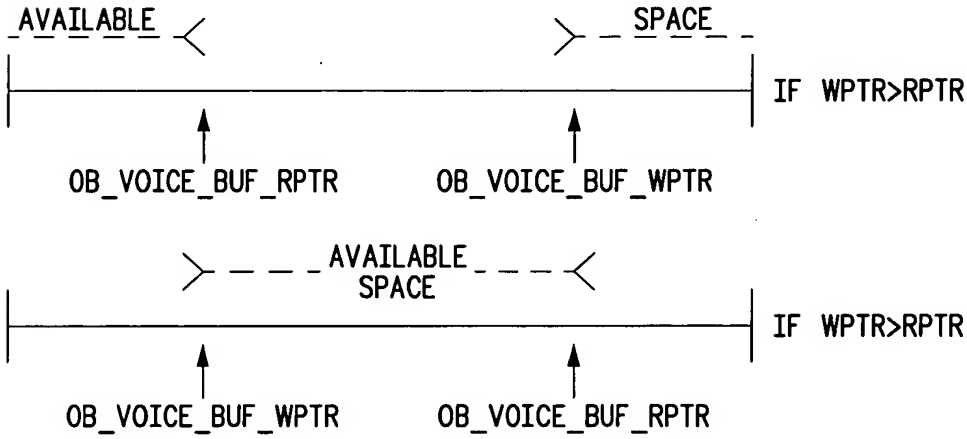
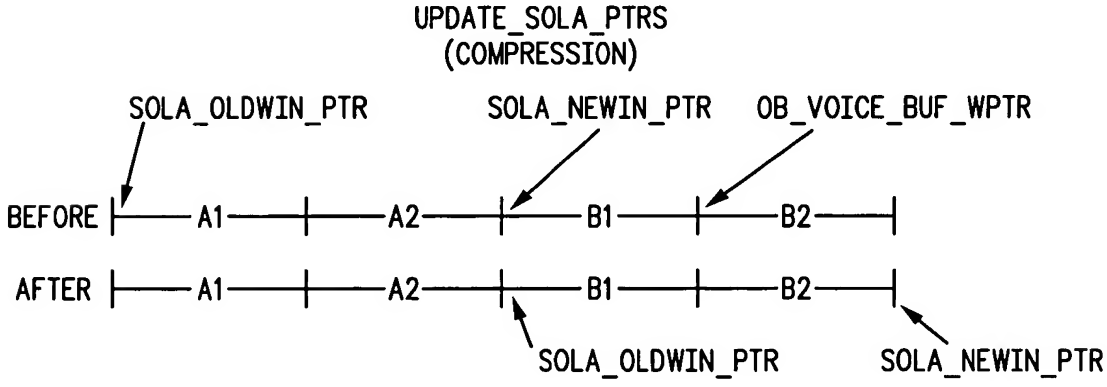


FIG. 13



OUTBOUND AUDIO BUFFER 1024 SAMPLES

1 FRAME IS [A1 A2] OF LENGTH N WHERE A1 IS THE FIRST HALFFRAME

FIG. 14

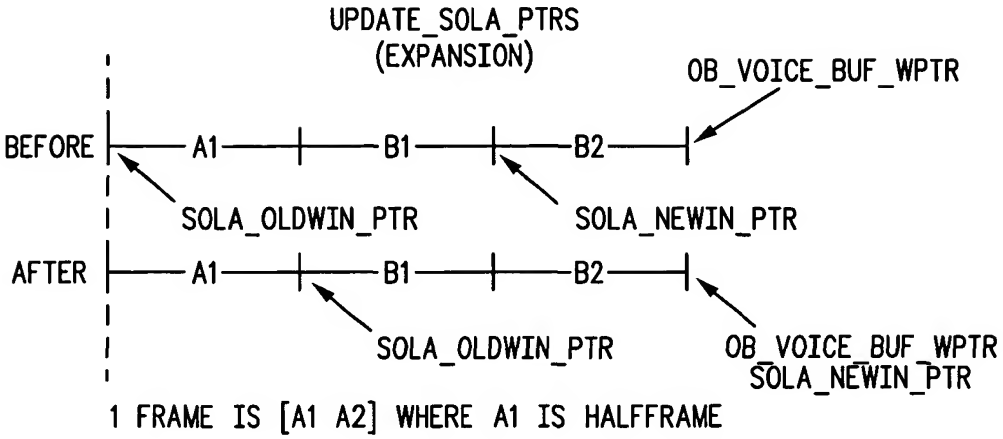

UPDATE_SOLA_PTRS
(EXPANSION)
OB_VOICE_BUF_WPTR
BEFORE
A1
B1
B2
SOLA_OLDWIN_PTR
SOLA_NEWIN_PTR
AFTER
A1
B1
B2
SOLA_OLDWIN_PTR
OB_VOICE_BUF_WPTR
SOLA_NEWIN_PTR
1 FRAME IS [A1 A2] WHERE A1 IS HALFFRAME

FIG. 15

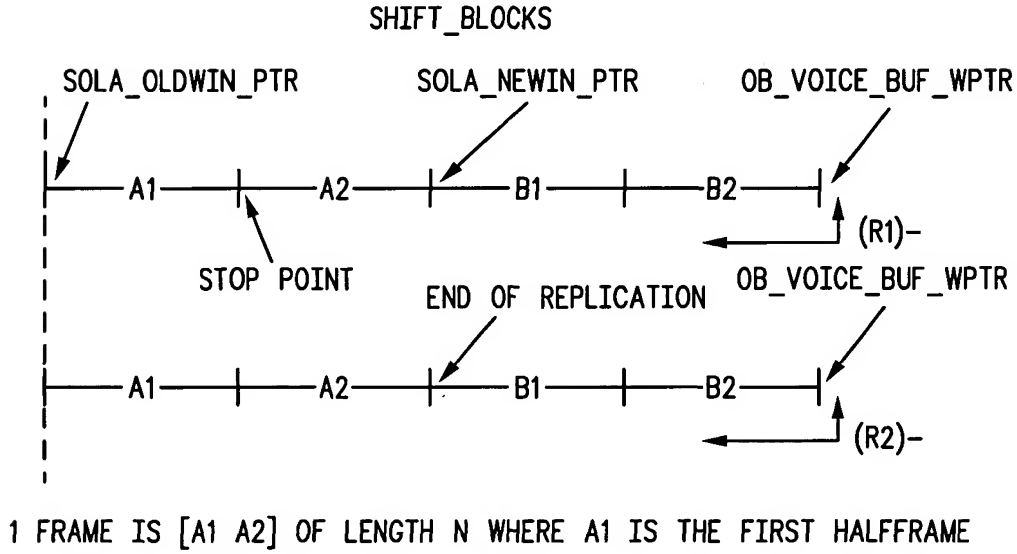

SHIFT_BLOCKS
SOLA_OLDWIN_PTR
SOLA_NEWIN_PTR
OB_VOICE_BUF_WPTR
A1
A2
B1
B2
(R1)-
STOP POINT
END OF REPLICATION
OB_VOICE_BUF_WPTR
A1
A2
B1
B2
(R2)-
1 FRAME IS [A1 A2] OF LENGTH N WHERE A1 IS THE FIRST HALFFRAME

FIG. 16

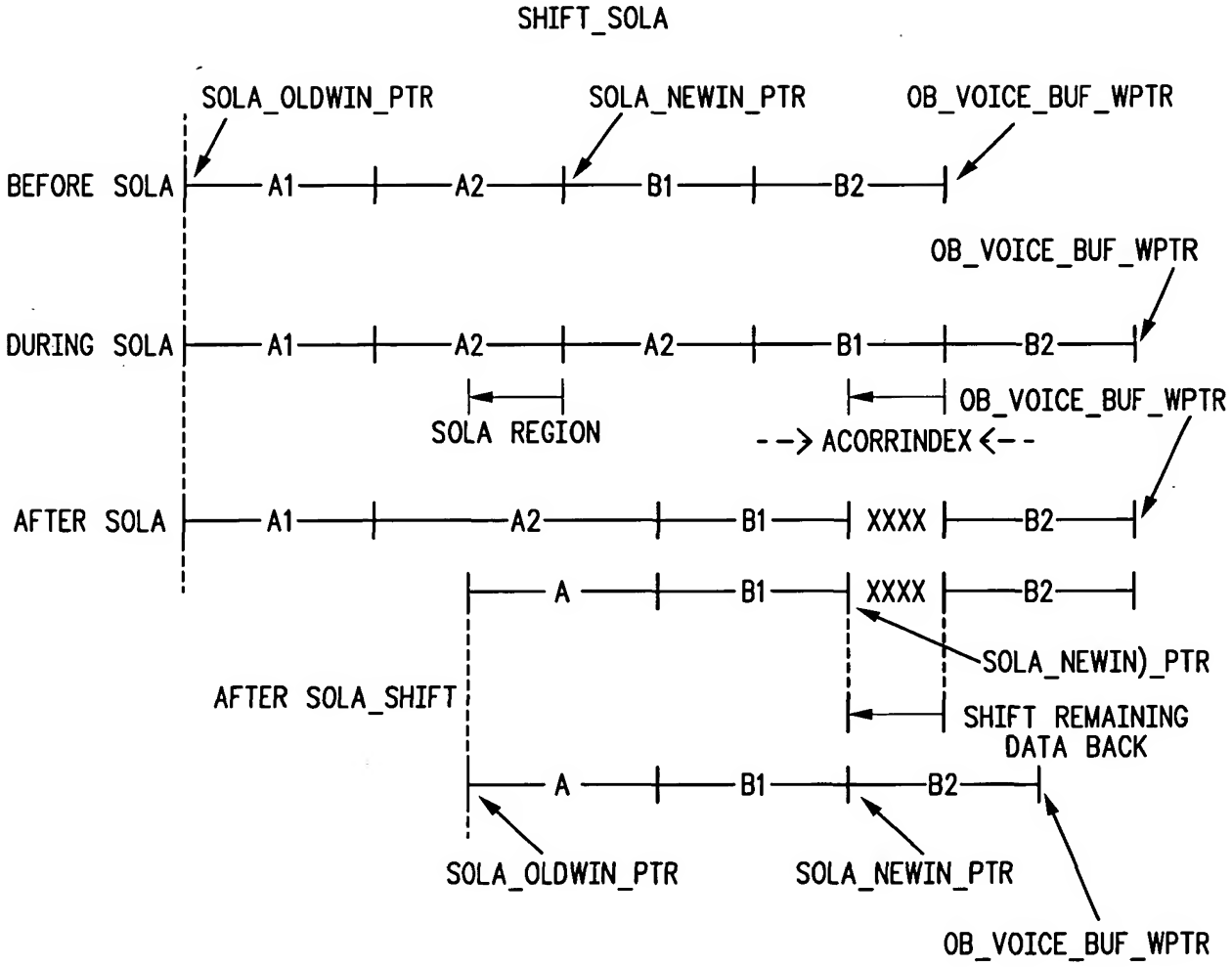

SHIFT_SOLA
SOLA_OLDWIN_PTR
SOLA_NEWIN_PTR
OB_VOICE_BUF_WPTR
BEFORE SOLA
A1
A2
B1
B2
OB_VOICE_BUF_WPTR
DURING SOLA
A1
A2
A2
B1
B2
SOLA REGION
OB_VOICE_BUF_WPTR
--> ACORRINDEX <--
AFTER SOLA
A1
A2
B1
XXXX
B2
A
B1
XXXX
B2
SOLA_NEWIN)_PTR
AFTER SOLA_SHIFT
SHIFT REMAINING
DATA BACK
A
B1
B2
SOLA_OLDWIN_PTR
SOLA_NEWIN_PTR
OB_VOICE_BUF_WPTR

FIG. 17

```
%-----------------------------------------------
% TEMPLATE FOR ASSEMBLY CODING
%
% Implement temporal compression sola algorithm straight up
% It simply processes frame by on a real time basis
%
% CALLS SOLA 0
%-----------------------------------------------

clear
path(path,'d;\sola\asm_sola3');
x=hexreadib('scott1.io',0);


global y oldwin;
global winlen winstep cc_range;
global runindex1 runindex2;
global cc_valsI cc_vals2

Nframes=13;
compress=0;

   % convert time of window length to number of samples winlen = 180;

      if(compress)
      winstep=winlen;
      nwins=Nframes;
      cc_range=winlen/2;   %conpression
      else
      winstep=winlen/2;
    nwins=2*Nframes;
       cc_range=winstep/2;   %expansion (==winlen/4)
      end
   % compute total munber of windows in input vector

   % cc_range is used to set cross-correlation detection range

 oldwin=x(1:winlen);
 y=oldwin;
 runindex1=1;
 runindex2=winlen;

 for i=2:nwins
   begin=(i-1)*winstep+1;
   frame=x(begin; begin+wilen-1);
   y=sola(frame);
 end
```

FIG. 18

```
%-------------------------------------------
%
% This is an implementation of the synchronized overlap and ad method (SOLA)
% for time scale compression
%
% SPECIFICALLY FOR USE AS A REAL TIME APP WHICH ACCEPTS 1 FRAME PER
% X-should be one frame of speech only
% Y-storage be one frame of speech only
% cf-compression factor
%
%-------------------------------------------

function [y]=sola(newwin)

global y;
global winlen winstep cc_range;
global runindex1 runindex2;
global cc_vals1 cc_vals2

   % compute the cross-corelation
   the_corr=xcorr(oldwin, newwin);

   [maxcc,indx]= max(the_corr(:cc_range));

   runindex2=winlen+runindex1 -1;   %always end of old frame

   runindex1=runindex2-indx+1;
   grad=[1indx]'/(indx+1);

   % sola region
   y(runindex1:runindex2)=y(runindex1;runindex2).*flipud(grad)...
   +newwin(:indx).*grad;

   % append remainder of new frame
   :runindex1+winlen-1=newwin((indx+1):winlen);
   oldwin=y(runindex1:runindex]+winlen-1);
```

FIG. 19

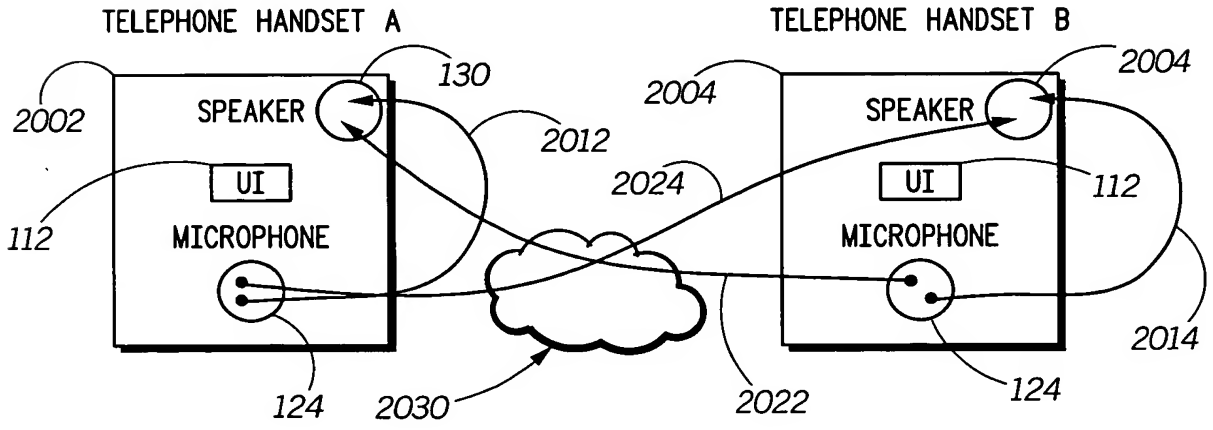


FIG. 20